MCEN5125 – Optimal Design of Engineering Systems
Instructor: Assistant Professor G. Subbarayan

PROJECT #1:
Algorithms for Optimization of Unconstrained Engineering Systems

by

Joseph P. Kubitschek
4 October 1998

ABSTRACT
Various unconstrained optimization algorithms were investigated for the purpose of obtaining an improved understanding of optimization techniques. FORTRAN code for the Method of Steepest Descent and Newton's Method was developed independently to obtain the minimum values of two test problems, Rosenbrock's function and the tip loaded cantilever beam maximum deflection function. In addition, the refined IMSL routines, DUMING and DUMIAH (Quasi-Newton Methods) were used to evaluate the same functions for comparison purposes. The results have been tabulated and are presented as objective-function contour plots with superimposed optimization routine trajectories and were compared in each case from the standpoints of accuracy and efficiency.

INTRODUCTION

The purpose of this project is to develop an improved understanding of various algorithms used for optimization of unconstrained engineering systems. Three primary algorithms that are commonly employed include the Method of Steepest Descent, Newton's Method, and the Quasi-Newton Method. FORTRAN code for the Method of Steepest Descent and Newton's Method was developed independently and used to compute the minimum of Rosenbrock's function (a common test problem for optimization algorithms) and the maximum deflection of a tip loaded cantilever beam. In addition, refined IMSL routines for optimization were used for comparison purposes. This project has been divided into seven tasks that include:

1. Derivation of displacement field for a tip loaded cantilever beam.
2. Independent development of FORTRAN code for the Method of Steepest Descent and Newton's Method.
3. Review of usage for IMSL library FORTRAN routines.
4. Solutions to minimum of Rosenbrock's function and the maximum deflection of a tip loaded cantilever beam using the Method of Steepest Descent, Newton's Method, and IMSL routines DUMING and DUMIAH.
5. Derivation of the number of function evaluations necessary to calculate one forward difference gradient and one forward difference hessian.
6. Comparison of algorithms and determination of best algorithm for optimizing unconstrained engineering systems.
7. Implementation of line search procedure in the Method of Steepest Descent and Newton's Method and comparison of accuracy and efficiency with and without line search.

Rosenbrock's function is a "narrow-valley" function given as

$$f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

The traditional starting point of $\mathbf{x} = (-1.2,1.0)$ was used during this project. The minimum value of the function is $f^*(\mathbf{x}) = 0$ and occurs at $\mathbf{x^*} = (1.0,1.0)$.

The maximum deflection function for a tip loaded cantilever beam is given as

$$f(x_1,x_2) = 12x_1^2 + 4x_2^2 - 12x_1x_2 + 2x_1.$$

In this case, the traditional starting point of $\mathbf{x} = (-1.0,-2.0)$ is also used during this project. The minimum value of the function is $f^*(\mathbf{x}) = -1/3$ and occurs at $\mathbf{x^*} = (-1/3.-1/2)$.

Solution Techniques

Task 1**:**
See appendix A.

Task 2:
The Method of Steepest Descent is a first order method that makes use of the gradient of the function and can be expressed, algorithmically, as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\mathbf{s},$$

where α is the step length and $\mathbf{s}$ is the search direction that is obtained as

$\mathbf{s} = -\nabla f(\mathbf{x})/\text{norm}[\nabla f(\mathbf{x})]$.

As can be seen, this algorithm makes use of the gradient in attaining the search direction. Similarly, Newton's method, which is a second order method, may be expressed algorithmically as

$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\mathbf{s}$.

In this case,

$\mathbf{s} = -\mathbf{H}^{-1}\nabla f(\mathbf{x}_k)$.

In contrast to steepest descent, Newton's Method makes use of the hessian as well as the gradient of the function. The Method of Steepest Descent and Newton's Method FORTRAN codes are included in appendix C as *stpdes.f* and *newmeth.f*, respectively. Both of these programs make use of subroutines that are coded by the user for minimization of the desired function. These subroutines consist of *fcn.f*, *grad.f*, and *hess.f*. The subroutine *fcn(n,x,f)* evaluates the objective function, returning f, given x(n), an n-dimensional vector, and n the dimension of x or number of independent variables. Similarly, the subroutine *grad(n,x,fgrad)* evaluates the gradient vector $\nabla f(x)$ of the objective function, returning fgrad(n), given x(n) and n. Finally, the subroutine *hess(n,x,h,ldh)* evaluates the hessian of the objective function, $\nabla^2 f(x)$, a nxn dimensional array, returning h(n,n) and ldh the leading dimension (i.e. row dimension), given x(n), and n. Each subroutine was coded for Rosenbrock's function and the cantilever beam deflection function for solution using *stpdes.f*, *newmeth.f* and IMSL routines DUMING and DUMIAH. However, the Method of Steepest Descent does not make use of the hessian, hence it was precluded from use in the program *stpdes.f*.

Task 3:
IMSL routine usage was obtained by logging onto *eddie* and accessing *pubbs* bulletin #2. This usage information was printed for future reference.

Task 4:
The test problems, Rosebrock's function and the cantilever beam deflection function, were coded as the previously discussed subroutines and run using the Method of Steepest Descent, Newton's Method, and the IMSL routines DUMING and DUMIAH. The driver code; programs *rfunc1.f*, *cfunc1.f*, *rfunc2.f*, and *cfunc2.f* for optimization routines DUMING and DUMIAH, respectively are also included in appendix C. The results have been included as output in appendix D.

Task 5:
See appendix B.

The second step consisted of a comparative evaluation of the various optimization routines. For this purpose a performance index was developed and consists of the total number of function evaluations required per step divided by the total number of function evaluations required for convergence. This may be expressed mathematically as $\eta = f/F$, where f is the total number of function evaluations per step (i.e. one forward difference gradient and one forward difference hessian) as derived above as $n^2/2+n+1$, and F is the total number of function evaluations required for convergence to some level of accuracy that is specified *apriori* (i.e. eps=1.0e-8 in FORTRAN codes). Thus, a performance index η=1.0 represents an optimally efficient case as convergence is

obtained in a single step. Alternatively, as $\eta \rightarrow 0$, the algorithm is decreasingly efficient since the total number of function evaluations is large in comparison to that required in a single step. However, it is important to note that this holds in the case of convergence. If convergence is not attained, then the performance index, in this form, provides no indication of the accuracy, another important characteristics of a sound optimization routine.

The number of function, gradient, and hessian evaluations were tabulated for each code in attaining the minimum along with **x**, the function values at convergence, the number of iterations, the convergence criteria, and the performance index.

Finally, the trajectories generated by each algorithm as it approached the minimum are presented along with the objective function contour plots for each case. The bold face entries in the tabulated results represent those cases for which trajectory plots were generated. These results are included in appendix E and discussed in the following results and discussion section.

Task 6:
Comparison of algorithms was achieved using the results of the task 5 and is included in the following results and discussion section.

Task 7:
Includes the implementation of a line search procedure for the Method of Steepest Descent algorithm (*stpdesm.f*) and Newton's Method algorithm (*newmthm.f*) was unsuccessful.

RESULTS AND DISCUSSION

Tabulated Results

For this investigation, the convergence criteria was specified *aproiri* as eps=10e-8. In addition, 1,000 iterations were also specified as a cut-off under conditions where convergence could not be achieved. However, this limit was pushed in the case of the Method of Steepest Descent in an unsuccessful attempt to improve accuracy and attain convergence. Table 1 represents the results for the minimization of Rosenbrock's function using the Method of Steepest Descent without a line search procedure. Various step-length, α, values were run to investigate the convergence characteristics of this algorithm. It is apparent that, in the case of Rosenbock's function the Method of Steepest Descent has difficulty converging within the specified cut-off criteria of 1000 iterations. Only after reducing the step length to α=0.001 with 100,000 iterations was third order accuracy achieved. This represents a relatively large computation cost and poor performance parameters, η, as compared with other methods investigated here.

Table 1. – Computed results for Rosenbrock's function using Method of Steepest Descent without line search algorithm.

α	f*	X_1	x_2	#fcn	#grad	#it	η
0.10	**0.75522657**	**0.53557**	**0.21338**	**1**	**1,001**	**1,000**	**0.005**
0.25	1.29090982	0.11440	0.08427	1	1,001	1,000	0.005
0.50	**15.00144417**	**0.03590**	**0.37641**	**1**	**1,001**	**1,000**	**0.005**
0.10	0.75522657	053560	0.21341	1	10,001	10,000	0.0005
0.05	0.26121312	0.78626	0.57178	1	1,001	1,000	0.005
0.01	**0.02804301**	**0.86175**	**0.75206**	**1**	**1,001**	**1,000**	**0.005**
0.001	2.57755850	-0.60354	0.37214	1	1,001	1,000	0.005
0.001	**0.00012524**	**0.99998**	**0.99883**	**1**	**10,001**	**10,000**	**0.0005**
0.001	0.00012510	1.00035	0.99958	1	100,001	100,000	0.00005

Table 2 represents the tabulated results for the minimization of cantilever beam function using the Method of Steepest Descent. Again, some difficulty converging to the solution is evident. Only after reducing the step-length to α=0.01 was third order accuracy of the minimum achieved in 1,000 iterations. For no case was the performance parameter greater than 0.005.

Table 2. – Computed results for Cantilever Beam function using Method of Steepest Descent without line search algorithm.

α	f*	X_1	X_2	#fcn	#grad	#it	η
0.10	**-0.30285400**	**-0.29387**	**-0.52112**	**1**	**1,001**	**1,000**	**0.005**
0.25	-0.09564938	-0.44355	-0.44102	1	1,001	1,000	0.005
0.50	**0.77703785**	**-0.57154**	**-0.37251**	**1**	**1,001**	**1,000**	**0.005**
0.10	-0.30285400	-0.29387	-0.52112	1	10,001	10,000	0.0005
0.05	-0.33105206	-0.32254	-0.50578	1	1,001	1,000	0.005
0.01	**-0.33329073**	**-0.33186**	**-0.50079**	**1**	**1,001**	**1,000**	**0.005**
0.001	-0.33310455	-0.32991	-0.50183	1	1,001	1,000	0.005

Newton's Method represents much improved results as given in tables 3-4. In all step-length cases for both functions investigated, convergence to the solution was achieved. Varying α between 0.1 and 1.0 demonstrates the improved efficiency as the number of iterations is improved from 277 to 7 for minimization of Rosenbrock's function; and improvement from 190 to 1 for the cantilever beam function, respectively. This case for which α=1.0 represents a performance parameter of η=0.417 for Rosenbrock's function and η=1.250 for the cantilever beam function, which is remarkable in comparison with the Method of Steepest Descent. It should be noted that the performance index for the cantilever beam function using Newton's is greater than 1.0. This is due to the fact that finite difference techniques were not used in these algorithms because the exact values for the gradient and hessian were explicitly known. Thus, all of the performance indices presented here are slightly higher than those that would be obtained had the gradient and hessian of the functions not be explicit (i.e. These algorithms would have required more function evaluations).

Table 3. – Computed results for Rosenbrock's function using Newton's Method without line search algorithm.

α	f*	x_1	x_2	#fcn	#grad	#hess	#it	η
0.10	**.00000000**	**1.00000**	**1.00000**	**1**	**278**	**277**	**277**	**0.009**
0.25	**.00000000**	**1.00000**	**1.00000**	**1**	**115**	**114**	**114**	**0.022**
0.50	**.00000000**	**1.00000**	**1.00000**	**1**	**57**	**56**	**56**	**0.044**
0.75	**.00000000**	**1.00000**	**1.00000**	**1**	**36**	**35**	**35**	**0.069**
1.00	**.00000000**	**1.00000**	**1.00000**	**1**	**6**	**5**	**7**	**0.417**

Table 4. – Computed results for Cantilever Beam function using Newton's Method without line search algorithm.

α	f*	x_1	x_2	# fcn	#grad	#hess	#it	η
0.10	**-.33333333**	**-.33333**	**-.50000**	**1**	**191**	**190**	**190**	**0.013**
0.25	**-.33333333**	**-.33333**	**-.50000**	**1**	**71**	**70**	**70**	**0.035**
0.50	**-.33333333**	**-.33333**	**-.50000**	**1**	**30**	**29**	**29**	**0.083**
0.75	**-.33333333**	**-.33333**	**-.50000**	**1**	**16**	**15**	**15**	**0.156**
1.00	**-.33333333**	**-.33333**	**-.50000**	**1**	**2**	**1**	**1**	**1.250**

The primary advantage of using the IMSL routines is in their level of refinement. As can be seen in table 5, convergence is attained in a relatively small number of iterations. The fact that these results represent improvement, even though the performance parameters are not necessarily higher, is realized in consideration of the step-length refinement. Obviously the IMSL routine make use of some sort of line search procedure to optimized the step length at each iteration. This precludes the need to investigate various values of α in obtaining a more efficient convergence as was done with the results in tables 1-4.

Table 5. - IMSL Routines (DUMING and DUMIAH) used to computed minimum of Rosenbrock's function and beam deflection function.

DUMING								
	f*	**x_1**	**x_2**	**#fcn**	**#grad**		**#it**	**η**
Rosenbrock's funct.	.00000000	.99999997	.99999993	32	23	N/A	19	0.091
Cantilever Beam	-.3333333	-.3333333	-.5000000	8	5	N/A	4	0.385
DUMIAH								
	f*	**x_1**	**x_2**	**#fcn**	**#grad**	**#hess**	**#it**	**η**
Rosenbrock's funct.	.00000000	.99999991	.99999982	31	22	21	21	0.068
Cantilever Beam	-.3333333	-.3333333	-.5000000	5	2	1	1	0.625

Trajectory Results

The trajectories for each algorithm provide a visual indication of the path to convergence. These trajectories have been superimposed over the contour plots for Rosebrock's function and the cantilever beam function and are included as appendix E. Figures 1 and 2 represent the contour plot of Rosenbrock's function and the cantilever beam function, respectively, in the independent variable space. Figures 3-6 represent the superimposed trajectories of the Method of Steepest Descent for α = 0.001, 0.01, 0.1, and 0.5, respectively. It can be seen that these trajectories follow a path along the valley and only in the case of α = 0.001 is convergence nearly achieved. However, this is at the expense of a large number of iterations (i.e. 10,000). It is also important to note that a small-scale oscillation about the path occurs, a feature that cannot be seen because only every 100^{th} data point was plotted. As α is increased, it is apparent that the termination point becomes further away from the optimum solution. Figures 7-9 represent the Method of Steepest Descent applied to the cantilever beam function for α = 0.01, 0.1, and 0.5 respectively. In this case the trajectories take a more direct approach to the optimum solution. However, again convergence is not attained in 1,000 iterations. Figures 10-14 represent Newton's Method applied to Rosenbrock's function for the full range of step-length values investigated (i.e. α = 0.1, 0.25, 0.5, 0.75, ad 1.0). In each of these cases, convergence is obtained as this algorithm takes a more direct approach (i.e. without the small-scale oscillations) to the optimum solution. Furthermore, it can be seen that for α = 1.0 the most efficient case exists. Similarly, figures 15-19 show direct convergence to the optimum solution for the cantilever beam function. Again, α = 1.0 represent the most efficient case as convergence is achieved in a single step. It is important to note that this will not always be true, because with larger values of α comes increase risk of over-shooting the optimum solution, a potential problem that the line search procedure handles

nicely. Finally, figures 20-23 represent the trajectories obtained from the IMSL routines, DUMING and DUMIAH. Figure 20 (i.e. the DUMING algorithm trajectory) shows a large first step in approaching the minimum of Rosenbrock's function, followed by a reversal in direction and ultimately convergence. This algorithm, unlike all the other algorithms does not follow the valley to the solution during the initial iteration. Figure 21 shows relatively fast convergence in the case of the cantilever beam function using DUMING as the solution was obtained in only two iterations. Unlike figure 20, figure 22 illustrates the nature of the DUMIAH routine in attaining convergence. Similar to Newton's method, this algorithm trajectory follows the valley during the path to convergence. However, it appears to be somewhat side-tracked at times, a likely feature of the line search procedure that is being used. Although it does not represent a direct path, it is certainly more efficient than the haphazard selection of α that was done for Newton's Method during this investigation. Finally, figure 23 shows, similar to Newton's Method with $\alpha = 1.0$, direct convergence in a single step. It is important to note that a greater number of function evaluations were required in this case. However, in the absence of knowing α *apriori*, this algorithm is likely to be more efficient.

CONCLUSIONS

- Newton's Method represents an improved algorithm in comparison with the Method of Steepest Descent. For the two test problems investigated, convergence is gauranteed with Newton's method, regardless of the step-length, α. However, the IMSL routines, DUMING and DUMIAH are likely more efficient due to their level of refinement in optimizing α at each step and the fact that they make use of Newton's method without the inherent deficiencies (i.e. Quasi-Newton Methods)

- An indicator of performance can be derived from the minimum number of function evaluations required at each step as compared with the total number of function evaluations required for convergence. This was obtained as $\eta$and was used to compare each of the algorithms investigated from the standpoint of efficiency. Although η is not a direct indicator of accuracy, it does provide some indication with respect to how quickly an algorithm converges. If the algorithm does not converge then η provides no indication of relative accuracy and becomes an indicator only of efficiency in achieved a pre-specified level of accuracy.

- The trajectories presented for each case provide insight into the algorithmic characteristics. For the case of the Method of Steepest Descent, it is apparent that the algorithm is less efficient as small-scale oscillations exist, even thought the trajectory tends to follow the Rosenbrock's function valley. In the case of Newton's method, a more direct approach to the solution exist which lends to it's improved efficiency.

- The IMSL routines, DUMING and DUMIAH likely represent the best efficiency due to their "Quasi-Newton" nature. In both cases of test problems, convergence to the solution was achieved quickly without prior knowledge of the step-length, α, a characteristics indicative of the level of refinement and a good optimization algorithm.

REFERENCES

1. Haftka, R.T., and Z. Gurdal, Elements of Structural Optimization, 3rd Ed., Kluwer Academic Publishers, 1992.

2. Chapra, S.C., and R.P. Canale, Introduction to Computing for Engineers, McGraw-Hill Book Co., 1986.

3. C.F. Gerald, and P.O. Wheatley, Applied Numerical Analysis, 4th Ed., Addison-Wesley Publishing Co., 1989.

APPENDIX A – Task #1: Displacement Field Derivation

This task called for derivation of the displacement field (eq. 4.2.15, Haftka and Gurdal[1]) which is given as

$$v(\xi) = [(1-3\xi^2+2\xi^3) \;\; l(\xi-2\xi^2+\xi^3) \;\; (3\xi^2-2\xi^3) \;\; l(-\xi^2+\xi^3)]\,[v_1 \;\; \theta_1 \;\; v_2 \;\; \theta_2]^T, \quad (1)$$

starting with the general solution for the displacement field given as

$$v(\xi) = a_0 + a_1\xi + a_2\xi^2 + a_3\xi^3, \quad (2)$$

where a_0, a_1, a_2, a_3 are constant coefficients, and $\xi = x/l$ is the position along the length of the beam. The approach to this derivation consists of first applying the boundary conditions to obtain four equations in four unknowns (i.e. the coefficients). The system of equations can then be solved using Gauss elimination to obtain expressions for the coefficients in terms of v_1, θ_1, v_2, and θ_2. These expressions may then be substituted into the general solution (eq. 2) to obtain the result, eq. 4.2.15. The boundary conditions are

Fixed end (at $\xi = 0$ or $x = 0$):

1. $v(0) = a_0 = v_1$. (3)
2. $dv/d\xi(0) = a_1 = \theta_1 l$. (4)

Free end (at $\xi = 1$ or $x = l$):

3. $v(1) = a_0 + a_1 + a_2 + a_3 = v_2$. (5)
4. $dv/d\xi(0) = a_1 + 2a_2 + 3a_3 = \theta_2 l$. (6)

These four boundary conditions produce the following system of equations that may be used to solve for the coefficients:

$$\begin{aligned} &a_0 = v_1, \\ &a_1 = \theta_1 l, \\ &a_0 + a_1 + a_2 + a_3 = v_2, \\ &a_1 + 2a_2 + 3a_3 = \theta_2 l. \end{aligned} \quad (7)$$

Applying Gauss elimination gives expressions for the constant coefficients in terms of v_1, θ_1, v_2, and θ_2 as

$$\begin{aligned} &a_0 = v_1, \\ &a_1 = \theta_1 l, \\ &a_2 = -2\theta_1 l - 3v_1 + 3v_2 - \theta_2 l, \\ &a_3 = \theta_1 l + 2v_1 - 2v_2 + \theta_2 l. \end{aligned} \quad (8)$$

Finally, substitution into general solution (eq. 2) gives upon rearranging

$$v(\xi) = v_1(1-3\xi^2+2\xi^3) + \theta_1 l(\xi-2\xi^2+\xi^3) + v_2(3\xi^2-2\xi^3) + \theta_2 l(-\xi^2+\xi^3). \quad (9)$$

Eq. 9 is exactly the result given as eq. 4.2.15 from which the tip loaded cantilever beam deflection objective function is obtained.

APPENDIX B – Task #5: Function Evaluations and Performance Index

The first step in this task was to derive the number of function evaluations necessary to calculate one forward difference gradient and one forward difference hessian. The function expansion about a point, δxi, may be expressed as a Taylor series

$$f(\mathbf{x} + \delta x_i) = f(\mathbf{x}) + \nabla f^T(\mathbf{x})\delta x_i + \text{H.O.T.}'^{s}$$

Neglecting higher order terms and rearranging gives

$$\nabla f^T(\mathbf{x})\delta x_i = [f(\mathbf{x} + \delta x_i) - f(\mathbf{x})]/\delta x_i.$$

This expression represents one forward difference gradient. Now, f(**x**) requires one function evaluation and f(**x** + δx_i) requires n function evaluations corresponding with i = 1,2,…,n. Thus, the total number of function evaluations required for the calculation of one forward difference gradient is n+1. Similarly, the hessian can be expressed using the previously obtained gradient as

$$\mathbf{H} = \nabla^2 f^T(\mathbf{x}) = \partial/\partial x_j\{[f(\mathbf{x} + \delta x_i) - f(\mathbf{x})]/\delta x_i\}.$$

Differentiating and rearranging gives

$$\nabla^2 f^T(\mathbf{x}) = [f(\mathbf{x} + \delta x_i + \delta x_j) - 2f(\mathbf{x} + \delta x_i) + f(\mathbf{x})]/\ \delta x_i \delta x_j.$$

This expression represents one forward difference hessian for which f(**x**) requires one function evaluation, f(**x** + δx_i) requires n functions evaluations corresponding with i=1,2,…,n and f(**x** + δx_i + δx_j) requires $n^2/2$ function evaluations corresponding with i=1,2,…,n and j=1,2,…,n for i≠j. Thus a total of $n^2/2+n+1$ function evaluations are required to compute one forward difference hessian.

APPENDIX C – FORTRAN CODE

1.) ROSENBROCK'S FUNCTION MINIMIZATION

a.) Steepest Descent Method

```
      PROGRAM stpdesr
c     Computes the minimum value of a multi-variate function using Method of Steepest Descent.
      INTEGER n,count,fcount,gcount,hcount
      PARAMETER (n=2,alpha=0.1,eps=1.0e-8)
      DOUBLE PRECISION x(n),fgrad(n),fgrads,f,h(n,n),ldh
c
      x(1)=-1.2
      x(2)=1.0
      fcount=0
      gcount=0
      hcount=0
      count=0
10    gcount=gcount+1
      call gradr(n,x,fgrad)
      fgrads=sqrt(fgrad(1)**2+fgrad(2)**2)
      if (fgrads.gt.eps) then
        x(1)=x(1)-alpha*fgrad(1)/fgrads
        x(2)=x(2)-alpha*fgrad(2)/fgrads
c     write output x(1) & x(2) at each iteration for redirect
c     to generate trajectory plots.
        write(*,*) x(1),x(2)
        if (count.lt.1000) then
          count=count+1
          go to 10
         else
          go to 20
        endif
      else
        go to 20
      endif
20    fcount=fcount+1
      call fcnr(n,x,f)
      open(unit=45,file='stpdesr.out',status='unknown')
      write(45,*) ' Minimum value of function is '
      write(45,30) f
      write(45,*) 'count   fcount   gcount      x1       x2'
      write (45,40) count,fcount,gcount,x(1),x(2)
30    format(5x,f12.8)
40    format(1x,i5,5x,i5,5x,i5,2x,f7.5,2x,f7.5)
      END
```

b.) Newton's Method

```
      PROGRAM newmethr
c     Computes the minimum value of a multi-variate function using Newton's Method.
      INTEGER n,i,count,fcount,gcount,hcount
      PARAMETER (n=2,alpha=0.1,eps=1.0e-8)
      DOUBLE PRECISION x(n),fgrad(n),fgrads,f,h(n,n),ldh,s(n),deth
c
      x(1)=-1.2
      x(2)=1.0
      fcount=0
      gcount=0
      hcount=0
      count=0
10    gcount=gcount+1
      call gradr(n,x,fgrad)
      fgrads=sqrt(fgrad(1)**2+fgrad(2)**2)
      if (fgrads.gt.eps) then
       hcount=hcount+1
       call hessr(n,x,h,ldh)
c     Cramer's Rule
       deth=h(1,1)*h(2,2)-h(1,2)*h(2,1)
       s(1)=(h(2,2)*fgrad(1)-h(1,2)*fgrad(2))/deth
       s(2)=(h(1,1)*fgrad(2)-h(2,1)*fgrad(1))/deth
       x(1)=x(1)-alpha*s(1)
       x(2)=x(2)-alpha*s(2)
c     write output x(1) & x(2) at each iteration for redirect
c     to generate trajectory plots.
       write(*,*) x(1),x(2)
       if (count.lt.1000) then
        count=count+1
        go to 10
       else
        go to 20
       endif
      else
       go to 20
      endif
20    fcount=fcount+1
      call fcnr(n,x,f)
      open(unit=45,file='newmthr.out',status='unknown')
      write(45,*) ' Minimum value of function is '
      write(45,30) f
      write(45,*) 'count   fcount   gcount      x1       x2'
      write (45,40) count,fcount,gcount,x(1),x(2)
30    format(5x,f12.8)
40    format(1x,i5,5x,i5,5x,i5,2x,f7.5,2x,f7.5)
      END
```

c.) Function subroutine

```
      SUBROUTINE fcnr(n,x,f)
c     Evaluates the function specified (Rosenbrock's funct.)
      INTEGER n
      DOUBLE PRECISION x(n),f
c
      f=100.*(x(2)-x(1)**2)**2+(1.-x(1))**2
      return
      END
```

d.) Gradient subroutine

```
      SUBROUTINE gradr(n,x,fgrad)
c     Evaluates the gradient of the function specified (Rosenbrock's funct.)
      INTEGER n
      DOUBLE PRECISION x(n),fgrad(n)
c
      fgrad(1)=400.*x(1)**3-400.*x(1)*x(2)+2.*x(1)-2.
      fgrad(2)=-200.*x(1)**2+200.*x(2)
      return
      END
```

e.) Hessian Subroutine

```
      SUBROUTINE hessr(n,x,h,ldh)
c     Evaluates the hessian of the function specified (Rosenbrock's funct.)
      INTEGER n
      DOUBLE PRECISION x(n),h(n,n)
c
      h(1,1)=-400.*x(2)+1200.*x(1)**2+2.
      h(2,2)=200.
      h(1,2)=-400.*x(1)
      h(2,1)=-400.*x(1)
      return
      END
```

2.) CANTILEVER BEAM DEFLECTION

a.) Steepest Descent Method

```
      PROGRAM stpdesc
c     Computes the minimum value of a multi-variate function using Method of Steepest Descent.
      INTEGER n,count,fcount,gcount,hcount
      PARAMETER (n=2,alpha=0.1,eps=1.0e-8)
      DOUBLE PRECISION x(n),fgrad(n),fgrads,f,h(n,n),ldh
c
      x(1)=-1.0
      x(2)=-1.2
      fcount=0
      gcount=0
      hcount=0
      count=0
10    gcount=gcount+1
      call gradc(n,x,fgrad)
      fgrads=sqrt(fgrad(1)**2+fgrad(2)**2)
      if (fgrads.gt.eps) then
        x(1)=x(1)-alpha*fgrad(1)/fgrads
        x(2)=x(2)-alpha*fgrad(2)/fgrads
c     write output x(1) & x(2) at each iteration for redirect
c     to generate trajectory plots.
        write(*,*) x(1),x(2)
        if (count.lt.1000) then
          count=count+1
          go to 10
         else
          go to 20
        endif
      else
        go to 20
      endif
20    fcount=fcount+1
      call fcnc(n,x,f)
      open(unit=45,file='stpdesc.out',status='unknown')
      write(45,*) 'Maximum deflection is '
      write(45,30) f
      write(45,*) 'count   fcount   gcount      x1       x2'
      write (45,40) count,fcount,gcount,x(1),x(2)
30    format(5x,f12.8)
40    format(1x,i5,5x,i5,5x,i5,2x,f7.5,2x,f7.5)
      END
```

b.) Newton's Method

```
      PROGRAM newmethc
c     Computes the minimum value of a multi-variate function using Newton's Method.
      INTEGER n,i,count,fcount,gcount,hcount
      PARAMETER (n=2,alpha=0.1,eps=1.0e-8)
      DOUBLE PRECISION x(n),fgrad(n),fgrads,f,h(n,n),ldh,s(n),deth
c
      x(1)=-1.0
      x(2)=-2.0
      fcount=0
      gcount=0
      hcount=0
      count=0
10    gcount=gcount+1
      call gradc(n,x,fgrad)
      fgrads=sqrt(fgrad(1)**2+fgrad(2)**2)
      if (fgrads.gt.eps) then
       hcount=hcount+1
       call hessc(n,x,h,ldh)
c     Cramer's Rule
       deth=h(1,1)*h(2,2)-h(1,2)*h(2,1)
       s(1)=(h(2,2)*fgrad(1)-h(1,2)*fgrad(2))/deth
       s(2)=(h(1,1)*fgrad(2)-h(2,1)*fgrad(1))/deth
       x(1)=x(1)-alpha*s(1)
       x(2)=x(2)-alpha*s(2)
c     write output x(1) & x(2) at each iteration for redirect
c     to generate trajectory plots.
       write(*,*) x(1),x(2)
       if (count.lt.1000) then
        count=count+1
        go to 10
       else
        go to 20
       endif
      else
       go to 20
      endif
20    fcount=fcount+1
      call fcnc(n,x,f)
      open(unit=45,file='newmthc.out',status='unknown')
      write(45,*) 'Maximum deflection is'
      write(45,30) f
      write(45,*) 'count   fcount   gcount      x1       x2'
      write (45,40) count,fcount,gcount,x(1),x(2)
30    format(5x,f12.8)
40    format(1x,i5,5x,i5,5x,i5,2x,f7.5,2x,f7.5)
      END
```

c.) Function subroutine

```
      SUBROUTINE fcnc(n,x,f)
c     Evaluates the function specified (cantilever beam)
      INTEGER n
      DOUBLE PRECISION x(n),f
c
      f=12.*x(1)**2+4.*x(2)**2-12.*x(1)*x(2)+2.*x(1)
      return
      END
```

d.) Gradient subroutine

```
      SUBROUTINE gradc(n,x,fgrad)
c     Evaluates the gradient of the function specified (cantilever beam)
      INTEGER n
      DOUBLE PRECISION x(n),fgrad(n)
c
      fgrad(1)=24.*x(1)-12.*x(2)+2.
      fgrad(2)=8.*x(2)-12.*x(1)
      return
      END
```

e.) Hessian Subroutine

```
      SUBROUTINE hessc(n,x,h,ldh)
c     Evaluates the hessian of the function specified (cantilever beam)
      INTEGER n
      DOUBLE PRECISION x(n),h(n,n)
c
      h(1,1)=24.
      h(2,2)=8.
      h(1,2)=-12.
      h(2,1)=-12.
      return
      END
```

IMSL ROUTINE DRIVER CODE

1.) DUMING:

a.) Rosenbrock's Function

```
      PROGRAM rfunc1
c     Computes the minimum value of Rosenbrock's function using refined
c     IMSL routine DUMING.
      INTEGER n,iparam(7)
      PARAMETER (n=2)
      DOUBLE PRECISION x(n),f,grad(n),xguess(n),xscale(n),fscale,
     & rparam(7)
      EXTERNAL fcnr,gradr,DUMING
c
      xguess(1)=-1.2
      xguess(2)=1.0
      xscale(1)=1.0
      xscale(2)=1.0
      fscale=1.0
      iparam(1)=0.0
      call DUMING(fcnr,gradr,n,xguess,xscale,fscale,iparam,rparam,x,f)
c     Print results
      open(unit=45,file='dumr1.out',status='unknown')
      write(45,*)'The solution is          x1          x2'
      write(45,10)x
      write(45,*)'The minimum value is'
      write(45,20)f
      write(45,*)'The number of iterations ='
      write(45,30)iparam(3)
      write(45,*)'The number of function evaluations ='
      write(45,30)iparam(4)
      write(45,*)'The number of gradient evaluations ='
      write(45,30)iparam(5)
10    format(20x,f12.8,5x,f12.8)
20    format(5x,f12.8)
30    format(5x,i12)
      END
```

b.) Cantilever Beam Deflection Function

```
      PROGRAM cfunc1
c     Computes the minimum value of beam deflection function using refined
c     IMSL routine DUMING.
      INTEGER n,iparam(7)
      PARAMETER (n=2)
      DOUBLE PRECISION x(n),f,grad(n),xguess(n),xscale(n),fscale,
     & rparam(7)
      EXTERNAL fcnc,gradc,DUMING
c
      xguess(1)=-1.2
      xguess(2)=1.0
      xscale(1)=1.0
      xscale(2)=1.0
      fscale=1.0
      iparam(1)=0.0
      call DUMING(fcnc,gradc,n,xguess,xscale,fscale,iparam,rparam,x,f)
c     Print results
      open(unit=45,file='dumc1.out',status='unknown')
      write(45,*)'The solution is          x1          x2'
      write(45,10)x
      write(45,*)'The minimum value is'
      write(45,20)f
      write(45,*)'The number of iterations ='
      write(45,30)iparam(3)
      write(45,*)'The number of function evaluations ='
      write(45,30)iparam(4)
      write(45,*)'The number of gradient evaluations ='
      write(45,30)iparam(5)
10    format(20x,f12.8,5x,f12.8)
20    format(5x,f12.8)
30    format(5x,i12)
      END
```

2.) DUMIAH:

a.) Rosenbrock's Function

```
      PROGRAM rfunc2
c     Computes the minimum of Rosnbrock's function using refined
c     IMSL routine DUMIAH.
      INTEGER n,iparam(7)
      PARAMETER (n=2)
      DOUBLE PRECISION x(n),f,grad(n),h(n,n),ldh,xguess(n),xscale(n),
     &  fscale,rparam(7)
      EXTERNAL fcnr,gradr,hessr,DUMIAH
c
      xguess(1)=-1.0
      xguess(2)=-2.0
      xscale(1)=1.0
      xscale(2)=1.0
      fscale=1.0
      iparam(1)=0.0
      call DUMIAH(fcnr,gradr,hessr,n,xguess,xscale,fscale,iparam,rparam,
     &  x,f)
c     Print results
      open(unit=45,file='dumr2.out',status='unknown')
      write(45,*)'The solution is          x1          x2'
      write(45,10)x
      write(45,*)'The minimum value is'
      write(45,20)f
      write(45,*)'The number of iterations ='
      write(45,30)iparam(3)
      write(45,*)'The number of function evaluations ='
      write(45,30)iparam(4)
      write(45,*)'The number of gradient evaluations ='
      write(45,30)iparam(5)
      write(45,*)'The number of hessian evaluations ='
      write(45,30)iparam(7)
10    format(20x,f12.8,5x,f12.8)
20    format(5x,f12.8)
30    format(5x,i12)
      END
```

b.) Cantilever Beam Deflection Function

```
      PROGRAM cfunc2
c     Computes the maximum deflection of cantilever beam using refined
c     IMSL routine DUMIAH.
      INTEGER n,iparam(7)
      PARAMETER (n=2)
      DOUBLE PRECISION x(n),f,grad(n),h(n,n),ldh,xguess(n),xscale(n),
     &  fscale,rparam(7)
      EXTERNAL fcnc,gradc,hessc,DUMIAH
c
      xguess(1)=-1.0
      xguess(2)=-2.0
      xscale(1)=1.0
      xscale(2)=1.0
      fscale=1.0
      iparam(1)=0.0
      call DUMIAH(fcnc,gradc,hessc,n,xguess,xscale,fscale,iparam,rparam,
     &  x,f)
c     Print results
      open(unit=45,file='dumc2.out',status='unknown')
      write(45,*)'The solution is          x1          x2'
      write(45,10)x
      write(45,*)'The minimum value is'
      write(45,20)f
      write(45,*)'The number of iterations ='
      write(45,30)iparam(3)
      write(45,*)'The number of function evaluations ='
      write(45,30)iparam(4)
      write(45,*)'The number of gradient evaluations ='
      write(45,30)iparam(5)
      write(45,*)'The number of hessian evaluations ='
      write(45,30)iparam(7)
10    format(20x,f12.8,5x,f12.8)
20    format(5x,f12.8)
30    format(5x,i12)
      END
```

MATLAB PLOT CODE:

1.) Rosenbrock's Function

```
%Contour plot of Rosenbrock's function.
x=(-2:.1:2);
y=(-2:.1:3);
for n=1:41
  for m=1:51
    F(n,m)=100.*(y(m)-x(n)^2)^2+(1-x(n))^2;
    X(n,m)=x(n);
     Y(n,m)=y(m);
  end
end
contour(X,Y,F,50);
title('ROSENBROCKS FUNCTION CONTOUR PLOT');
xlabel('x1');
ylabel('x2');
print -dps rosplt.ps;
```

Typical Trajectory Plot Files

a.) fn = sdtrpltr.m

```
%Plot of Rosenbrock's function trajectory.
load sdtrajr.out
x=(-2:.1:2);
y=(-4:.1:3);
for n=1:41
  for m=1:71
    F(n,m)=100.*(y(m)-x(n)^2)^2+(1-x(n))^2;
    X(n,m)=x(n);
     Y(n,m)=y(m);
  end
end
traj1(1)=-1.2
traj2(1)=1.0
for n=200:100:10000
  i=n/100;
  traj1(i)=sdtrajr(n,1)
  traj2(i)=sdtrajr(n,2)
end
contour(X,Y,F,50);
title('STEEPEST DESCENT TRAJECTORY (alpha=0.001)');
xlabel('x1');
ylabel('x2');
hold on;
plot(traj1,traj2,'-',traj1,traj2,'*');
print -dps sdtrajr.ps
```

b.) fn = nmtrpltr.m

```
%Plot of Rosenbrock's function trajectory.
load nmtrajr.out
x=(-2:.1:2);
y=(-4:.1:3);
for n=1:41
  for m=1:71
    F(n,m)=100.*(y(m)-x(n)^2)^2+(1-x(n))^2;
    X(n,m)=x(n);
     Y(n,m)=y(m);
```

```
  end
end
traj1(1)=-1.2
traj2(1)=1.0
for n=2:277
  traj1(n)=nmtrajr(n,1)
  traj2(n)=nmtrajr(n,2)
end
contour(X,Y,F,50);
title('NEWTONS METHOD TRAJECTORY (alpha=0.1)');
xlabel('x1');
ylabel('x2');
hold on;
plot(traj1,traj2,'-',traj1,traj2,'*');
print -dps nmtrajr.ps
```

2.) Cantilever Beam Function

```
Fn=cantplot.m
%Contour plot of Cantilever Beam function.
x=(-2:.1:2);
y=(-3:.1:3);
for n=1:41
  for m=1:61
    F(n,m)=12.*x(n)^2+4.*y(m)^2-12.*x(n)*y(m)+2.*x(n);
    X(n,m)=x(n);
     Y(n,m)=y(m);
  end
end
contour(X,Y,F,50);
title('CANTILEVER BEAM FUNCTION CONTOUR PLOT');
xlabel('x1');
ylabel('x2');
print -dps cantplt.ps;
grid on
```

Typical Trajectory Plot Files

a.) fn = sdtrpltc.m

```
%Plot of Cantilever Beam function trajectory.
load sdtrajc.out
x=(-2:.1:2);
y=(-4:.1:3);
for n=1:41
  for m=1:71
    F(n,m)=12.*x(n)^2+4.*y(m)^2-12.*x(n)*y(m)+2.*x(n);
    X(n,m)=x(n);
     Y(n,m)=y(m);
  end
end
traj1(1)=-1.0
traj2(1)=-2.0
for n=20:10:1000
  i=n/10;
  traj1(i)=sdtrajc(n,1)
  traj2(i)=sdtrajc(n,2)
end
contour(X,Y,F,50);
title('STEEPEST DESCENT TRAJECTORY (alpha=0.01)');
xlabel('x1');
ylabel('x2');
hold on;
plot(traj1,traj2,'-',traj1,traj2,'*');
print -dps sdtrajc.ps
```

b.) fn = nmtrpltc.m

```
%Plot of Cantilever Beam function trajectory.
load nmtrajc.out
x=(-2:.1:2);
y=(-4:.1:3);
for n=1:41
  for m=1:71
    F(n,m)=12.*x(n)^2+4.*y(m)^2-12.*x(n)*y(m)+2.*x(n);
    X(n,m)=x(n);
```

```
      Y(n,m)=y(m);
   end
end
for n=1:2
   traj1(n)=nmtrajc(n,1)
   traj2(n)=nmtrajc(n,2)
end
contour(X,Y,F,50);
title('NEWTONS METHOD TRAJECTORY (alpha=1.0)');
xlabel('x1');
ylabel('x2');
hold on;
plot(traj1,traj2,'-',traj1,traj2,'*');
print -dps nmtrajc.ps
```

APPENDIX D – OUTPUT

METHOD OF STEEPEST DESCENT:

1.) Rosenbrock's Function Minimization

α =0.1

fn=stpdesr.out
Minimum value of function is
.75522657

count	fcount	gcount	x1	x2
1000	1	1001	.53557	.21338

α =0.25

fn=stpdesr2.out
Minimum value of function is
1.29090982

count	fcount	gcount	x1	x2
1000	1	1001	.11440	.08427

α =0.5

fn=stpdesr3.out
Minimum value of function is
15.00144417

count	fcount	gcount	x1	x2
1000	1	1001	.03590	.37641

α =0.1, count=10,000

fn=stpdesr4.out
Minimum value of function is
.75522657

count	fcount	gcount	x1	x2
10000	1	10001	.53560	.21341

α =0.05, count=1,000

fn=stpdesr5.out
Minimum value of function is
.26121312

count	fcount	gcount	x1	x2
1000	1	1001	.78626	.57178

α =0.01, count =1,000

fn=stpdesr6.out
Minimum value of function is
.02804301

count	fcount	gcount	x1	x2
1000	1	1001	.86175	.75206

α =0.001, count =1,000

fn=stpdesr7.out
Minimum value of function is
2.57755850

count	fcount	gcount	x1	x2
1000	1	1001	-.60354	.37214

α =0.001, count =10,000

fn=stpdesr7.out

Minimum value of function is

.00012524

count	fcount	gcount	x1	x2
10000	1	10001	.99998	.99883

α =0.001, count =100,000

fn=stpdesr7.out

Minimum value of function is

.00012510

count	fcount	gcount	x1	x2
100001	1	100001	1.00035	.99958

2.) Cantilever Beam Deflection Function

α =0.1

fn=stpdesc.out
Maximum deflection is
-.30285400

count	fcount	gcount	x1	x2
1000	1	1001	-.29387	-.52112

α =0.25

fn=stpdesc2.out
Maximum deflection is
-.09564938

count	fcount	gcount	x1	x2
1000	1	1001	-.44355	-.44102

α =0.5

fn=stpdesc3.out
Maximum deflection is
.77703785

count	fcount	gcount	x1	x2
1000	1	1001	-.57154	-.37251

α =0.1, count=10,000

fn=stpdesc4.out
Maximum deflection is
-.30285400

count	fcount	gcount	x1	x2
10000	1	10001	-.29387	-.52112

α =0.05, count=1,000

fn=stpdesc5.out
Maximum deflection is
-.33105206

count	fcount	gcount	x1	x2
1000	1	1001	-.32254	-.50578

α =0.01, count =1,000

fn=stpdesc6.out
Maximum deflection is
-.33329073

count	fcount	gcount	x1	x2
1000	1	1001	-.33186	-.50079

α =0.001, count =1,000

fn=stpdesc7.out
Maximum deflection is
-.33310455

count	fcount	gcount	x1	x2
1000	1	1001	-.32991	-.50183

NEWTON'S METHOD:

1.) Rosenbrock's Function Minimization

α =0.1

fn=newmthr.out

Minimum value of function is

.00000000

count	fcount	gcount	x1	x2
277	1	278	1.00000	1.00000

α =0.25

fn=newmthr2.out

Minimum value of function is

.00000000

count	fcount	gcount	x1	x2
114	1	115	1.00000	1.00000

α =0.5

fn=newmthr3.out

Minimum value of function is

.00000000

count	fcount	gcount	x1	x2
56	1	57	1.00000	1.00000

α =0.75

fn=newmthr4.out

Minimum value of function is

.00000000

count	fcount	gcount	x1	x2
35	1	36	1.00000	1.00000

α =1.0

fn=newmthr5.out

Minimum value of function is

.00000000

count	fcount	gcount	x1	x2
6	1	7	1.00000	1.00000

2.) Cantilever Beam Deflection Function

α =0.1

fn=newmthc.out
Maximum deflection is
-.33333333

count	fcount	gcount	x1	x2
190	1	191	-.33333	-.50000

α =0.25

fn=newmthc2.out
Maximum deflection is
-.33333333

count	fcount	gcount	x1	x2
70	1	71	-.33333	-.50000

α =0.5

fn=newmthc3.out
Maximum deflection is
-.33333333

count	fcount	gcount	x1	x2
29	1	30	-.33333	-.50000

α =0.75

fn=newmthc4.out
Maximum deflection is
-.33333333

count	fcount	gcount	x1	x2
15	1	16	-.33333	-.50000

α =1.0

fn=newmthc5.out
Maximum deflection is
-.33333333

count	fcount	gcount	x1	x2
1	1	2	-.33333	-.50000

IMSL Routine DUMING:

1.) Rosenbrock's Function

```
fn=dumr1.out
The solution is         x1                  x2
                     .99999997        .99999993
 The minimum value is
   .00000000
 The number of iterations =
            19
 The number of function evaluations =
            32
 The number of gradient evaluations =
            23
```

2.) Cantilever Beam Deflection Function

```
fn=dumc1.out
The solution is         x1                  x2
                    -.33333333       -.50000000
 The minimum value is
   -.33333333
 The number of iterations =
             4
 The number of function evaluations =
             8
 The number of gradient evaluations =
             5
```

IMSL Routine DUMIAH:

1.) Rosenbrock's Function

fn=dumr2.out
The solution is x1 x2
.99999991 .99999982
The minimum value is
.00000000
The number of iterations =
21
The number of function evaluations =
31
The number of gradient evaluations =
22
The number of hessian evaluations =
21

2.) Cantilever Beam Deflection Function

fn=dumc2.out
The solution is x1 x2
-.33333333 -.50000000
The minimum value is
-.33333333
The number of iterations =
1
The number of function evaluations =
5
The number of gradient evaluations =
2
The number of hessian evaluations =

APPENDIX E – FIGURES

Figure 1. – Rosenbrock's function contour plot.

Figure 2. – Cantilever beam function contour plot.

Figure 3. – Trajectory of the Method of Steepest Descent used in obtaining the minimum of Rosenbrock's function, $\alpha = 0.001$.

Figure 4. – Trajectory of the Method of Steepest Descent used in obtaining the minimum of Rosenbrock's function, $\alpha = 0.01$.

Figure 5. – Trajectory of the Method of Steepest Descent used in obtaining the minimum of Rosenbrock's function, $\alpha = 0.1$.

Figure 6. – Trajectory of the Method of Steepest Descent used in obtaining the minimum of Rosenbrock's function, $\alpha = 0.5$.

Figure 7. – Trajectory of the Method of Steepest Descent used in obtaining the minimum of the cantilever beam function, $\alpha = 0.01$.

Figure 8. – Trajectory of the Method of Steepest Descent used in obtaining the minimum of the cantilever beam function, $\alpha = 0.1$.

Figure 9. – Trajectory of the Method of Steepest Descent used in obtaining the minimum of the cantilever beam function, $\alpha = 0.5$.

Figure 10. – Trajectory of Newton's Method used in obtaining the minimum of Rosenbrock's function, $\alpha = 0.1$.

Figure 11. – Trajectory of Newton's Method used in obtaining the minimum of Rosenbrock's function, $\alpha = 0.25$.

Figure 12. – Trajectory of Newton's Method used in obtaining the minimum of Rosenbrock's function, $\alpha = 0.5$.

Figure 13. – Trajectory of Newton's Method used in obtaining the minimum of Rosenbrock's function, $\alpha = 0.75$.

Figure 14. – Trajectory of Newton's Method used in obtaining the minimum of Rosenbrock's function, $\alpha = 1.0$.

Figure 15. – Trajectory of Newton's Method used in obtaining the minimum of the cantilever beam function, $\alpha = 0.1$.

Figure 16. – Trajectory of Newton's Method used in obtaining the minimum of the cantilever beam function, $\alpha = 0.25$.

Figure 17. – Trajectory of Newton's Method used in obtaining the minimum of the cantilever beam function, $\alpha = 0.5$.

Figure 18. – Trajectory of Newton's Method used in obtaining the minimum of the cantilever beam function, $\alpha = 0.75$.

Figure 19. – Trajectory of Newton's Method used in obtaining the minimum of the cantilever beam function, $\alpha = 1.0$.

Figure 20. – Trajectory of IMSL routine DUMING used in obtaining the minimum of Rosenbrock's function.

Figure 21. – Trajectory of IMSL routine DUMING used in obtaining the minimum of the cantilever beam function.

Figure 22. – Trajectory of IMSL routine DUMIAH used in obtaining the minimum of Rosenbrock's function.

Figure 23. – Trajectory of IMSL routine DUMIAH used in obtaining the minimum of the cantilever beam function.